PA1 Potion Shop Order Simulator

CIIC3015

PARTNER SELECTION DUE: Sep 19, 2025 11:59 PM

PROJECT DUE: Oct 3, 2025 11:59 PM

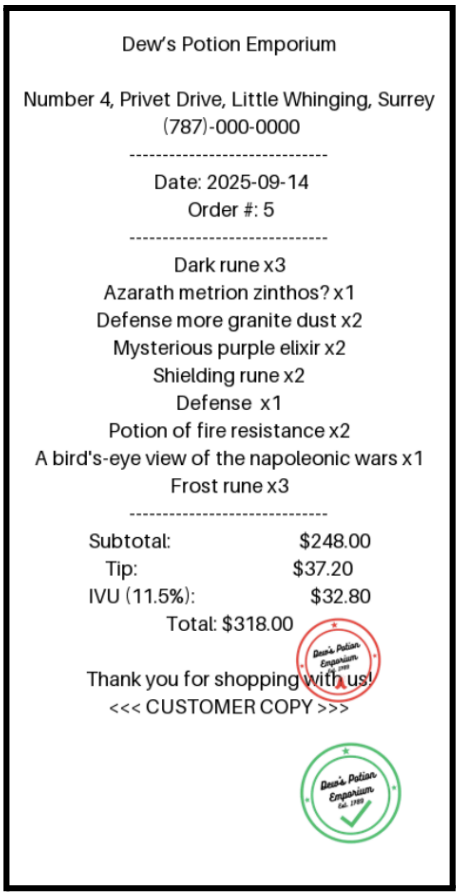
Partner Selection

Welcome to Project Assignment 1: Order Simulator! Before continuing, we highly recommend that you choose a partner for this project as soon as possible. You may select the same partner you worked with for Project Assignment 0 (Pong); however, if you do, you will not be allowed to work with the same person for future Project Assignments. Choose wisely! You have until **September 19, 2025** to select a partner and create a team on GitHub. After this date, you will no longer be able to create a team on GitHub. If you are having difficulty finding a partner, you can either post a message in the Find a Partner channel on the class Discord server or contact the TA in charge of projects.

Introduction

Dew's Potion Emporium is upgrading its ordering system, but it's currently incomplete. Your task is to develop a Python-based shop ordering system that enables users to browse the catalog, place customized orders, and calculate their total bill, including tax and tips.

This project will be implemented as a command-line application, where users interactively select catalog items, modify, and confirm their order. You will work in groups of two, utilizing GitHub Classroom for version control. Effect collaboration is key, each partner should contribute, and you must not work on the same laptop to ensure proper use of Git and commit tracking. Upon the completion of each order, a receipt image named `receipt.png` will be generated in your project files. This image has to be attached in the evaluation form. The receipt looks like this.



Setup

Accept the Assignment:

- Copy and paste the following link to accept or join PA0 https://classroom.github.com/a/wFvD4eB9 . One of the partners must create the group and the other partner must join it. DO NOT WORK ON THE SAME LAPTOP.

Packages

- For this project you will need Pillow. To install it you can run `pip install pillow` or in search the packages in pycharm as we did for in PA0 for PyGame

IMPORTANT

For all projects you will follow the same workflow as in PA0. Meaning that you should commit and push all tasks individually.

Tasks: Let's Tackle Those To-Do's!

Task 1: Potion Modification Menu 🧪

(Suggestion: Teammate A)

Implement the potion modification menu selection logic. Display available potion options, how ingredients for each, and allow the user to modify the selected item. The code must be done in the ModMenu() function.

This must prompt the user to select from at least 5 potions.

Potion	Ingredients
Healing	Phoenix Tear, Crimson Herb, Troll's Blood, Fiji Water, Dew's love ♡
Strength	Fire Salt, Ogre's Blood, Volcano Root, Dew's Fingernail?
Defense	Powdered Dragon Scale, Shell of a Koopa?, Granite Dust, Dew's Famous Petrified Moss
Clairvoyance	Owl's Feather, Powdered Crystal, Bottled Star, Moondrop Dew
Swiftness	Cheetah's Claw, Hummingbird Feather, Bottled Lightning, Summer Breeze, Dew's Ketchup?

Once the user has selected a potion it must show the user their selection and its ingredients. Then you must call the `ModifyItem` function. Next, prompt the user of the amount of the selected potion which they want and call the `AddItemToOrder` function.

Example:

```
Which Potion would you like you modify?
 (1.) Healing
 (2.) Strength
 (3.) Defense
 (4.) Clairvoyance
 (5.) Swiftness
< 2
> The Strength potion comes with: Fire Salt, Ogre's Blood, Volcano Root, Dew's Fingernail?
> Would you like to remove or add an ingredient?
 (1.) Remove
 (2.) Add
 (3.) Exit
< 2
> The ingredients are Fire Salt, Ogre's Blood, Volcano Root, Dew's Fingernail?
Which ingredient would you like to modify?
Fire Salt
> fire salt added.
> Would you like to remove or add an ingredient?
 (1.) Remove
 (2.) Add
 (3.) Exit
< 3
> How many would you like?
< 2
```

Task 2: ModifyItem 🛠️

(Suggestion: Teammate B)

Implement the logic that checks if the ingredient the user types exists in the ingredients string and keep track of any changes to summarize at the end. Use only if-else statements and basic string methods like .islower() and len() **no helper functions**. Make sure your code handles all inputs correctly, including invalid ones, organize your logic clearly!

Hint

You can compare the user's input to the ingredients list using string methods, and build a result string by adding phrases like "No <ingredient>" or "More <ingredient" depending on the action. Use simple if-else checks to decide what to add, and keep updating the string as the user makes multiple modifications. Check the image above to see it in action.

Task 3: Potion Menu 🪄

(Suggestion: Teammate B)

Implement the potion menu selection logic. Allow the user to choose the potion and ask for the quantity to add to the order. Implement your code in the PotionsMenu() function. This must at least prompt the user to choose between, Elixir of the Gilded Sun, Potion of flying, Mysterious Purple Elixir, Potion of Invisibility, Potion of Fire Resistance. Remember that to add the item to the order you must call the AddItemToOrder() function.

Example:

```
> Which potion would you like to order?
  (1.) Elixir of the Gilded Sun
  (2.) Potion of flying
  (3.) Mysterious Purple Elixir
  (4.) Potion of Invisibility
  (5.) Potion of Fire Resistance
< 3
> How many would you like?
< 7
```

Task 4: Books Menu 📚

This task is distributed into many functions:

- ### Task 4.1- Book Menu 📔

  (Suggestion: Teammate B)

  Implement the book's menu selection logic. Display available books and call the appropriate menu function for each book category. Do not prompt the user on how many books they want and add the item to order. This is done in the **BooksMenu()** function.

- ### Task 4.2 - History Menu 📦

  (Suggestion: Teammate A)

  Implement the history menu selection logic. Allow the user to choose a history book and return the selected item. The prompt of asking the user is already implemented! All you need to do here is handle the logic of the user's input. Implement your code in **HistoryMenu()**

- Task 4.3 - Spells Menu ✨

  (Suggestion: Teammate A)

  Implement the spell's menu selection logic. Allow the user to choose a spell book and return the selected item.Between the options it must at least have: Grimoire of the Veiled Path, The Book of Many Things, Azarath Metrion Zinthos? Implement your code in **SpellsMenu()**

- Task 4.4 - Other Menu 📦

  (Suggestion: Teammate B)

  Implement the coffee menu selection logic. Allow the user to choose a coffee type and return the selected item. The prompt of asking the user is already implemented! All you need to do here is handle the logic of the user's input. Implement your code in **OtherMenu()**

Example:

```
> Interested in my books? Which category would you like to browse?
  (1.) History
  (2.) Spells
  (3.) Other
< 1
> Which history book would you like to order?
  (1.) A Bird's-Eye View of the Napoleonic Wars
  (2.) Where the Silk Road Sings
  (3.) The Last Cemí
  {Not Available: The History of the Void Century}
< 2
```

Task 5: Runestone Menu 🪨

(Suggestion: Teammate A)

Implement the runestone menu selection logic. Allow the user to choose a runestone and return the selected item. Prompt the user for the amount and add it to the order. Between the potions it must at least have: Fire Rune, Frost Rune, Storm Rune, Earth Rune, Shielding Rune, Dark Rune, Light Rune. Implement your code in RunesMenu().

```
> Which runestone would you like to order?
  (1.) Fire Rune
  (2.) Frost Rune
  (3.) Storm Rune
  (4.) Earth Rune
  (5.) Shielding Rune
  (6.) Dark Rune
  (7.) Light Rune
< 3
> How many would you like?
< 2
```

Task 6: Sub Total 💰

(Suggestion: Teammate B)

Complete the logic for calculating the subtotal of the order. You can calculate the price based on the item and update the total accordingly. Add the price calculation logic for the different items, print the item details, and return the total. Your code must be implemented in the GetSubTotal(item, amount, total) function, which is called by GetSubTotalIterator. You don't need to worry about the iterator—just keep in mind that all the items are being passed to the function you are implementing. The prices are as follows:

Item	Price
Elixir of the Gilded Sun	13
Potion of Flying	10
Mysterious Purple Elixir	3
Potion of Invisibility	16
Potion of Fire Resistance	12
Any Modified Potion	23
A Bird's-Eye View of the Napoleonic Wars	7
Where the Silk Road Sings	7
The Last Cemí	5
Grimoire of the Veiled Path	10
The Book of Many Things	30
Azarath Metrion Zinthos?	5
1984	15
Pete the Cat: I Love My White Shoes	6
Don Quixote	8
One Hundred Years of Solitude	9
All Runes	20

A few notes:

- It doesn't matter what combinations and changes you make in the potion modification menu the resultant price has to always be 23
- All the runes are priced the same. One rune no matter which one it is will cost 20.

Task 7: Add Tip 📈

(Suggestion: Teammate A)

Implement the logic for adding a tip based on user input. The user should be able to choose between 10%, 15%, 20%, or 'No' tip. Calculate the tip and return the updated total. Additionally, create a tip_given variable that returns True or False depending on whether a tip was given.

Example:

```
Would you like to leave a tip?
1. 10%
2. 15%
3. 20%
4. No
< 1
TIP.........$4.00
```

Task 8: Add IVU 💸

(Suggestion: Teammate B)

Implement the logic for adding IVU (tax) to the total. The IVU rate is 11.5%. Return the updated total.

```
IVU.......$5.06
```

```
Your total is $49.06
```

Task 9: Receipt in Evaluation Form 🧾

(Must Be Done by Teammate A and B)

Once you have completed the implementation, please proceed to the evaluation form. You will be required to submit the receipt.png file that is generated after each completed order.

Please note: For the evaluation to be successful, the submitted receipt must include the tip stamp. This requires adding a tip during the checkout process.

Bonus 🚀

If you're aiming for bonus points, feel free to enhance your game by adding new features and making it exceptional. Bonus points range from 1-15, based on the TA's discretion.

Some ideas:

- Better UI
- Combo Deals
- Better Receipt Image Generation

Final Submission 🎯

To submit your project each team member must submit an evaluation form independently under PA1 in Moodle. One of the fields in the evaluation form will ask for a GitHub tag for your code. To generate a tag follow the following steps:

- Go to your repo and click on the `Releases` link usually found under the repository name.
- Click on the `Draft a new release` button.
- Under the `Tag version` field, enter the name of the tag you want to create (e.g v1.0.0).
- Fill in the release title and description (optional).
- Click on `Publish release` to create the tag and release.
- Enter the name of the tag in the evaluation form.
- Remember to submit the receipt.png!